Analysing the Evolution of Knowledge Graphs for the Purpose of Change Verification

AUTHOR(S):

Nishioka, Chifumi; Scherp, Ansgar

Analyzing the Evolution of Knowledge Graphs for the Purpose of Change Verification

Chifumi Nishioka
Kyoto University Library
Kyoto, Japan
Email: nishioka.chifumi.2c@kyoto-u.ac.jp

Ansgar Scherp
Kiel University and Leibniz Information Centre for Economics (ZBW)
Kiel, Germany
Email: asc@informatik.uni-kiel.de

***Abstract*—Knowledge graphs (KGs) are a core component of many web-based applications. KGs store information about entities such as persons and organizations. A central challenge is to keep KGs up-to-date while the entities in the real world continuously change. While the majority of the changes are correct, the KGs still receive erroneous changes due to vandalism and carelessness. Thus, change verification is required to ensure a quality of the information stored in the KG. Since manual change verification is labor intensive, different works have dealt with automatic change verification in the past. However, these works have not shed light on the evolutionary patterns of the KGs. Since the analysis of the evolution of social networks has contributed to link prediction between persons, we assume that the evolutionary patterns of KGs can contribute to the task of change verification. In this paper, we analyze the evolution of a KG, focusing on its topological features such as degree. The analysis reveals that the evolutionary patterns are similar to those of social networks. Subsequently, we develop classifiers that judge whether each incoming change is correct or incorrect. In the classifiers, we use a set of novel features, which originate from topological features of the KG. Finally, our experiments demonstrate that the novel features improve the verification performance. The results of this paper can contribute to making the KG editing process more efficient and reliable.**

I. Introduction

Knowledge graphs (KGs) store a large amount of entities such as persons, organizations, and locations and other facts about entities. They are widely used for many different applications including question answering [1] and document profiling [2]. However, it is difficult to keep the KGs up-to-date, since the real world continuously changes over time. In practice, a lot of human editors contribute to making changes on KGs such as Wikidata [3]. While the majority of the changes are correct, the KGs receive incorrect changes due to vandalism, carelessness, and misunderstanding by editors. In addition, since even automatically-created KGs such as DBpedia rely on Wikipedia infoboxes made by volunteer editors, it is not trivial for them to assess changes when importing the data. Thus, the task of change verification is demanding for KGs in general. In fact, Tanon et al. [4] argue that a significant increase in the amount of data needs to go along with either an increase in the number of KG editors or provision of tools to assist them to be more efficient. So far, only few works investigated automatic change verification [5], [6] using features computed from the content of the changes as well as the editors' history and expertise. Besides automatic change verification, also the research question of KG refinement has been studied [7]. KG refinement refers to the task to compute whether a new triple should be added or not, such as Dong et al. [8] who employ a path ranking algorithm and neural networks.

In this paper, we are to the best of our knowledge the first who analyze the evolution of a KG and make use of this novel information to predict whether a change on a KG is correct. We first analyze how KGs evolve over time. We apply methods known from analyzing the evolution of graphs such as social networks [9]. Subsequently, we develop classifiers for assessing whether changes on a KG are correct by exploiting the features from the KG evolution analysis. Different from existing change verification classifiers [5], [6], our classifier does not rely on editors' history. Thus, it can be applied to changes made by new editors and bots with no history of earlier changes.

As dataset, we use a series of snapshots of Wikidata over two years. Wikidata is one of the largest cross-domain KGs [3]. Our analysis reveals that Wikidata has similar evolultionary patterns to other graphs such as social networks. For example, Wikidata shows a preferential attachment model [10], i. e., entities that are frequently used as objects tend to be used more and more as objects. Our subsequent experiment on KG change verification demonstrates that our novel, evolution-based topological features are useful to automatically judge whether an incoming change is correct or incorrect. Particularly, the novel features help in correctly classifying changes where the object of the changed triple is a URI. This complements existing change verification methods, which perform better on assessing changes conducted on literals [6].

In summary, the contributions of this paper are: (i) we provide a precise analysis of the evolution of Wikidata, focusing on topological features, which were not investigated; (ii) we propose a classifier, which judges whether incoming changes are correct or not, using the novel features; and (iii) we demonstrate that the novel features and proposed classifier work well for the KG.

The remainder of this paper is organized as follows: In Section II, we review related works. Section III provides the problem definition. Section IV introduces the dataset of a KG, i .e., Wikidata, used in this paper. In Section V, we conduct the analysis of the evolution of the KG. Thereafter,

Section VI shows classifiers to judge the validity of each change. Section VII provides the results of the experiment. In Section VIII, we discuss our findings, before we conclude.

II. Related Work

In this section, we first review works that tackle the task of change verification on KGs. Subsequently, we look into the KG refinement. Finally, we summarize the works analyzing the evolution of KGs.

a) Change Verification on KGs: Tan et al. [5] employed three categories of features to classify changes made on Freebase into correct or incorrect ones. The features are computed from the content of triples, the editor history, and editor expertise. In terms of the content feature, they used only the predicate of the triple and ignored the subject and object. Editor history includes numbers of correct and incorrect edits in the past, as well as how long the editor have been working for Freebase. Editor expertise is how familiar an editor is with a specific domain. The authors employed logistic regression, Gradboost [11] and Perceptron [12] to classify changes using the features. Their experiment demonstrated that the classifier using all features perform best and the most effective feature is the triple feature. Heindorf et al. [6] proposed a set of 47 features to verify changes made for Wikidata. Their features can be categorized into two groups: content features and context features. Content features include textual features, triple features, and comment features. Context features contains editor features (e .g., their experience, their country), entity features (e .g., their popularity), and revision features. In their experiment, classifiers based on all features showed the highest performance. They observed that content features and context features contribute to improving precision and recall, respectively. Regarding classification algorithms, they obtain the best performance using a random forest [13] in combination with multiple-instance learning. While their features perform well, some of them such as comment features can be applied only to Wikidata.

b) KG Refinement: While the change verification aims at assessing the quality of each incoming change, the goal of the refinement of KGs is to add missing facts (i .e., completion) or identify erroneous facts (i .e., error detection) in a static KG [7]. Thus, the methods for the refinement are relevant for change verification. Paulheim [7] provides a survey of automatic KG refinement. In terms of completion, predicting a type or class for an entity is a common problem, which is investigated for over decades. Nickel et al. [14] used matrix factorization to predict entity types in YAGO. Regarding error detection in KGs, reasoning determines whether a given set of triples is free of contradictions [15]. However, it requires a rich ontology. Guéret et al. [16] used topological features such as degree, clustering coefficient, and centrality to define metrics for detecting wrong triples in KGs. They compared the actual distributions of the above mentioned metrics computed over the KG to the distributions that are ideally expected, e .g., a power-law distribution for the degree of entities. Then, they marked links that deviate from the ideal distributions as suspicious. Although there are many different methods for KG refinement, they are difficult to employ in an online change verification system. In addition, they have not exploited features computed from the evolution of KGs.

c) Evolution of KGs: Several works investigated the evolution of the Linked Open Data (LOD) cloud, which is a collection of different, interconnected KGs on the web. Käfer et al. [17] monitored $86,696$ RDF documents on the LOD cloud for 29 weeks and compared weekly snapshots of them. They observed that most RDF documents have not been changed. For the other RDF documents, the most dynamic elements are literals. Dividino et al. [18] analyzed the dynamics of KGs and have proposed a monotone, non-negative function to represent the dynamics of a set of triples as a single numerical value. We revealed representative patterns of entity changes using the DyLDO dataset [19]. In addition, we identified that entity changes have some periodicities.

III. Problem Definition

A change is represented by a 3-tuple, which is composed of a triple $\langle s, p, o\rangle$, a flag m, and a time stamp t. We compute a score for a change $(\langle s, p, o\rangle, m, t)$ on a KG G and classify it into correct or incorrect. A higher score indicates a change is likely incorrect. According to the standard RDF-based KG, a triple $\langle s, p, o\rangle$ consists of subject s, predicate p, and object o. Formally, we consider the sets of all possible resources R encoded as URIs for entities, classes, or predicates (i.e., properties in RDF) and literals L for values such as numbers and dates. In a triple $\langle s, p, o\rangle$, a subject $s \in R$ is a resource, a predicate $p \in R$ a resource, and an object $o \in R \cup L$ a resource or a literal. Naturally, KGs can be seen as a directed graph $G = (R \cup L; E)$, where each node from subject or object is either an entity $r \in R$ or a literal $l \in L$. The set of edges in the graph are described as $E = R \times P \times (R \cup L)$. A flag m indicates whether a triple is added ($m = 1$) or deleted ($m = -1$). Please note that a modification of a triple is considered as deletion with subsequent addition. A time stamp t shows a point in time at which a change is made. G_t and E_t indicate the KG and the set of edges at point in time t, respectively.

IV. Wikidata Dataset

As KG, we choose Wikidata, since it is widely used and covers different topics. Wikidata [3] is a cross-domain KG started in October 2012. It is maintained collaboratively by volunteer editors. One of its characteristics is that facts (i .e., statements along with the definition in Wikidata) may have quantifiers to store additional information such an information source (e .g., news article). However, we focus only on triples in this paper. Furthermore, due to the shutdown of Freebase, data in Freebase have been migrated into Wikidata [4]. Thus, Wikidata is expected to be used more widely. We use 25 snapshots of Wikidata from 04/20/2014 to 08/01/2016. We obtain the snapshots from the Wikidata RDF exports[1], where facts with quantifiers are converted into N-triples [20]. Thus,

[1] `wikidata-simple-statements.nt.gz` from each directory on http://tools.wmflabs.org/wikidata-exports/rdf/index.php?content=exports.php

an N-triple models the edges of the Knowledge Graphs following the definition above. The subject and object of the triple denote the two nodes that are connected via the edge, and the property of the triple determines the edge type. We filter out duplicate triples as well as triples that contain a blank node from the snapshots. Blank nodes may cause inconsistencies, since they may get different identifiers in different snapshots. On average, a snapshot contains $99,508,873.96$ triples (SD: $22,594,178.18$). There are $38,684,391$ unique URIs in all the snapshots. We obtain changes between two successive points in time by computing the difference between the two snapshots. The set of additions and deletions produced at point in time t are extracted from $E_t \setminus E_{t-1}$ and $E_{t-1} \setminus E_t$, respectively. In order to verify whether changes are correct or incorrect, we follow existing works [5], [21] and label a change as incorrect if it is reverted in around four weeks, otherwise correct. For example, $(\langle s,p,o\rangle, 1, t)$ is incorrect, if $(\langle s,p,o\rangle, -1, t+3\ weeks)$ is observed. Although we have 24 successive points in time, we may label changes made in 23 successive points in time. In each of the 23 successive points in time, on average $5,357,786.61$ triples are added, of which $333,331.09$ are incorrect. In terms of deleted triples, on average $1,997,224.91$ triples are deleted, of which $177,010.87$ are incorrect. Thus, on average 5.21% of added triples and 2.31% of deleted triples are incorrect.

V. Evolution of KGs at the Example of Wikidata

For the analysis of the evolution of the KG, we focus on the topological features, which have not been investigated before. We first look into the evolution of the number of nodes and edges, and examine whether the KG follows the densification power law [22], which is commonly observed in different graphs including social networks. If it follows, we may predict edges of the KG as it is done for social networks. Subsequently, we study from which kinds of nodes, edges are added or deleted. Thereafter, we investigate how the destination of added or deleted edges are selected. Finally, we explore how the predicate of added or deleted edges are chosen.

A. Global Properties

First, we investigate how global properties such as the number of nodes of the KG change over time. Figure 1 shows the numbers of nodes (i.e., entities and literals) and edges (i.e., triples) over time. If a graph follows the densification power law [22], it follows the relation $|E_t| \propto |R_t \cup L_t|^{\alpha}$, where α is an exponent that generally lies between 1 and 2. Figure 2 (a) depicts the relation between the number of nodes $|R_t \cup L_t|$ and the number of edges. Please note that both axes are in logarithmic scale. The plots are fit into a line well, but it does not perfect follow the densification power law, since $\alpha = 0.97$. The reason is that inherently a literal can hold only an in-degree of one (i.e., one edge). Thus, the graph goes to be sparse, as the number of literals increases. Therefore, we further investigate the densification power law excluding the influence by the literal nodes. To this end, we examine

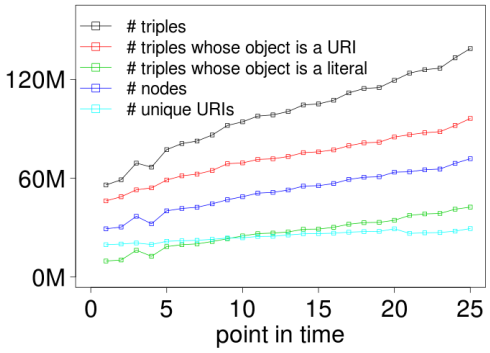
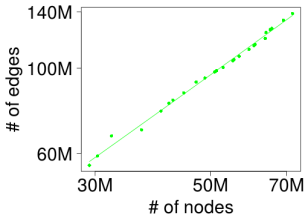
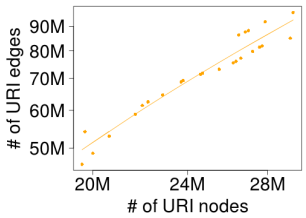


Fig. 1: Evolution of the number of nodes (i.e., entities and literals) and edges (i.e., triples) on Wikidata.



(a) The number of edges $|E_t|$ versus the number of nodes $|R_t \cup L_t|$



(b) The number of edges whose object is a URI $|(s,p,o) \in E_t : o \in R_t|$ versus the number of URI nodes $|R_t|$

Fig. 2: The number of edges versus the number of nodes. Both axes are in logarithmic scale.

the relation $|\langle s,p,o\rangle \in E_t : o \in R_t| \propto |R_t|$ as shown in Figure 2 (b). $|\langle s,p,o\rangle \in E_t : o \in R_t|$ denotes the number of edges whose object is a URI at a point in time t. In this case, the exponent is $\alpha = 1.56$, thus it follows the densification power law. We conclude that the connection among entities (i.e., a URI node) on the KG gets denser and denser over time, which is also observed in other graphs [22].

B. Edge Initiation

We investigate via which kinds of nodes, new edges are added or deleted. Especially, we investigate it from aspects: node degree, node age, and the last point in time at which a node was edited.

Node degree. We first explore the effect of a node degree on edge addition and edge deletion. Do rich nodes (i.e., nodes with high degree) bring more information to KGs? For the assessment, we compute in-degree as well as out-degree of subject nodes of added and deleted edges, before it is added or deleted. To this end, borrowing the definition from Leskovec et al. [9], we compute the average number of edges added or deleted by a node of a degree d as: $e(d,m) = \frac{\sum_{t\in T} |(\langle s,p,o\rangle,m,t):d_{t-1}(s)=d|}{\sum_{t\in T} |u:d_{t-1}(u)=d|}$.

In the equation, m is the flag indicating "added" or "deleted". u is an arbitrary node on the KG. $d(u)$ stands for the degree (either in-degree or out-degree) of a node u. Thus,

the numerator is the number of added or deleted edges made between $t-1$ and t whose degree of a subject is d. $e(d,m)$ is normalized by the number of nodes of degree d that exist just before this step. Figure 3 illustrates both in-degree and out-degree of subject nodes of added and deleted edges.

Node age. We examine the effect of a node age on edge addition and edge deletion. To this end, we compute $e(a,m)$, the average number of edges added or deleted by nodes of age a as: $e(a,m) = \sum_{t \in T} \frac{|(\langle s,p,o\rangle,m,t):t-t(s)=a|}{|u:t-t(u)=a|}$. $t(u)$ returns a point in time at which a node u is generated. Thus, the numerator counts the number of added or deleted edges where the age of the subject is a. It is normalized by the number of nodes whose age is a. Please note that we remove the nodes that appear at the first snapshot from this analysis to avoid truncation effects. We can see only that these nodes were generated between 10/30/2012 (i.e., the launch of Wikidata) and 04/20/2014, so their actual ages vary too much. Figure 4 plots the average number of added and deleted edges by a subject node whose age equals to a. Please note that the age is represented by points in time in Figure 4. Since the period between two successive points in time is approximately 36.05 days, nodes whose $a = 3$ is equal to 108.14 days old. As observed by Leskovec et al. [9], there is a small spike at $a = 0$ in Figure 4(a). The spike corresponds to nodes that generate information at the initial stage but never add further information to the KG. In addition, we observe a weak trend that the average number of added edges goes down as subject nodes get older. In terms of deleted edges, Figure 4(b) shows the trend that the number of deleted edges goes down, as well. It indicates that the older subjects are likely to be abandoned (i.e., no longer to be edited). Regarding incorrect changes, we do not observe a large difference in the curves of correct and incorrect ones.

Node last edit. In addition to node ages, we look into the influence by the period since the node is last edited. Do nodes that were recently edited add or delete more edges? We compute $e(b,m)$, the average number of edges added or deleted by the period b as: $e(b,m) = \sum_{t \in T} \frac{|(\langle s,p,o\rangle,m,t):t-tl(s)=b|}{|u:t-tl(s)=b|}$. $tl(u)$ returns the latest point in time at which a node u is edited. The numerator counts the number of added or deleted edged by a node which was last edited b points in time before. Then, it is normalized by the number of nodes that were lastly edited b points in time before. Figure 5 illustrates the result. Similar to the node age, we observe that both added and deleted edges decrease. The result indicates that the nodes will not be edited, if they are abandoned for a longer time.

C. Edge Destination Selection

We examine how edge destination of added and deleted edges are selected. We calculate again node degree, age, and last edit.

Node degree. We investigate the effect of node degrees on edge destination selection on the KG. The preferential attachment model [10] is known and observed in different social networks [9]. In this model, the likelihood of receiving new edges increases with the node degree. Do the KGs follow this model? In order to examine it, we compute $e(d,m)$, the average number of added and deleted edges with respect to different object degrees as: $e(d,m) = \sum_{t \in T} \frac{|\{(\langle s,p,o\rangle,m,t):d_{t-1}(o)=d\}|}{|\{u:d_{t-1}(u)=d\}|}$. The numerator is the number of added or deleted edges between $t-1$ and t whose degree of an object is d. It is normalized by the number of nodes of degree d that exist just before this step. Figure 6 shows the results. Please note that both axes are in logarithmic scale. As shown by Leskovec et al. [9], if a graph evolves randomly such as the Erdős-Rényi random network, the line is flat, as the destination node is chosen independently of its degree. In contrast, we observe the KG follows the preferential attachment model in terms of both correct and incorrect changes in Figure 6(a) and (b). In addition, we also observe the preferential attachment model in the deletions as Figure 6(e) and (f). In the in-degree of both additions and deletions, the incorrect changes fit to the relation $e(d,m) \propto d^{\alpha}$ better, since the distribution of the number of added or deleted edges at the high degrees is narrow. Referring to out-degree, we observe that the number of added and deleted edges follows the relation $e(d,m) \propto d^{\alpha}$ until the out-degree reaches 100. But the number of added and deleted edges is decreasing when the out-degree is over 100.

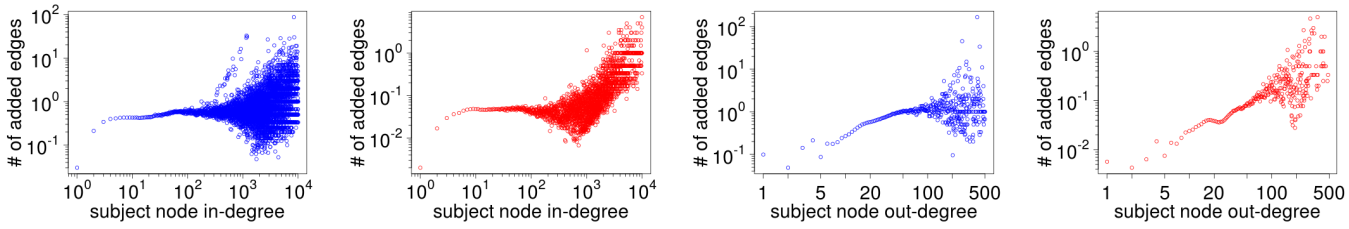
Node age. We examine the effect of a node age on addition and deletion of edges on the KG. Do older nodes receive more edges, since they are more experienced and known on the KG? We compute the average number of edges added or deleted by nodes of age a as: $e(a,m) = \sum_{t \in T} \frac{|\{(\langle s,p,o\rangle,m,t):t-t(o)=a\}|}{|\{u:t-t(u)=a\}|}$. Again, we remove the nodes that appear at the first snapshot from the analysis, as we do in Section V-B. Figure 7 plots the average number of added and deleted edges whose destination is a node of age a. Similar to Figure 4, new nodes receive more edges, although the scale of the y-axis is different. Regarding the correctness of changes, over 94% of additions are correct at each age except that only 69.03% are correct when $a = 0$. Thus, newer nodes more frequently receive incorrect changes. In addition, the probability of incorrect changes is also relatively high at $a = 0$ for the deletions.

Node last edit. Again, we look into the influence by the period since the node is last edited. We compute: $e(b,m) = \sum_{t \in T} \frac{|(\langle s,p,o\rangle,m,t):t-tl(o)=b|}{|u:t-tl(o)=b|}$. Figure 8 illustrates the result. Similar to the node age, the numbers of added and deleted edges are decreasing over time. In addition, the numbers of correct and incorrect changes are decreasing as well. These results indicate that nodes will not be edited if they are abandoned for a longer time.

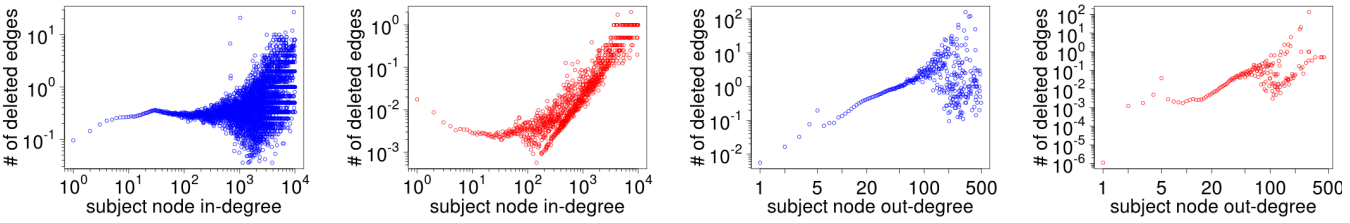
D. Relation Selection

Edges of KGs denote different relations, which are represented by predicates (e.g., `isMarriedTo`, `knows`). Thus, we analyze the effect of different predicates on the KG evolution.

Predicate age. As we did in Sections V-B and V-C, we examine the effect of a predicate age. Predicate age is defined as the length of time between the point of time it was first used and the current point of time. We compute $e(a)$, the

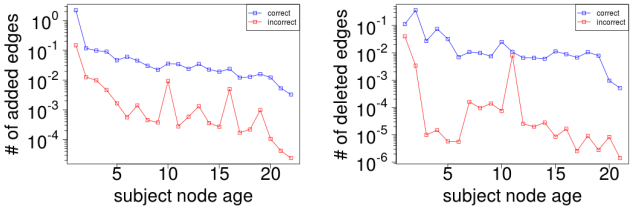


(a) Wikidata: In-degree of correct additions (b) Wikidata: In-degree of incorrect additions (c) Wikidata: Out-degree of correct additions (d) Wikidata: Out-degree of incorrect additions



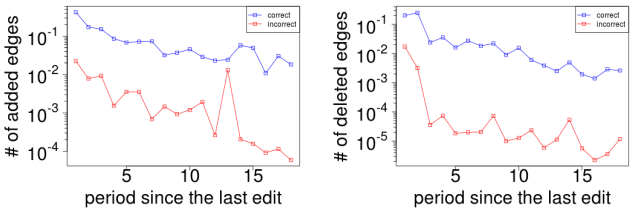
(e) Wikidata: In-degree of correct deletions (f) Wikidata: In-degree of incorrect deletions (g) Wikidata: Out-degree of correct deletions (h) Wikidata: Out-degree of incorrect deletions

Fig. 3: The average degree of subject nodes of added and deleted edges. The x-axis shows the average degree of nodes and the y-axis indicates the number of added or deleted edges. Both axes are in logarithmic scale.



(a) Wikidata: Added edges (b) Wikidata: Deleted edges

Fig. 4: The average number of added and deleted edges with a subject node of age a. The y-axis is in logarithmic scale.



(a) Wikidata: Added edges (b) Wikidata: Deleted edges

Fig. 5: The average number of added and deleted edges with a subject node that is last edited b points in time before. The y-axis is in logarithmic scale.

average number of added and deleted edges with a predicate whose age is a as: $e(a,m) = \sum_{t \in T} \frac{|\{\langle s,p,o\rangle,m,t) : t - t(p) = a\}|}{|q : t - t(q) = a|}$. q is an arbitrary predicate. The numerator counts the number of added or deleted edges with a predicate whose age is a, normalized by the number of predicates whose age is a. We remove the predicates that appear at the first snapshot to avoid truncation effects. Figure 9 illustrates the results with respect to the flags (i .e., added or deleted). We observe several peaks in both added and deleted edges. In Figure 9(a), we observe that added edges are decreasing. On the other hand, deleted edges are not decreasing in Figure 9(b).

Predicate last used. We look into the influence by the period since a predicate is used last. We compute: $e(b,m) = \sum_{t \in T} \frac{|\{\langle s,p,o\rangle,m,t) : t - tl(p) = b\}|}{|\{q : t - tl(q) = b\}|}$. Figure 10 plots the average number of added and deleted edges with respect to a predicate that was used b points in time before. We again observe that both added and deleted edges are decreasing. Incorrect changes likely use a predicate that was recently used.

VI. Change Verification on KGs

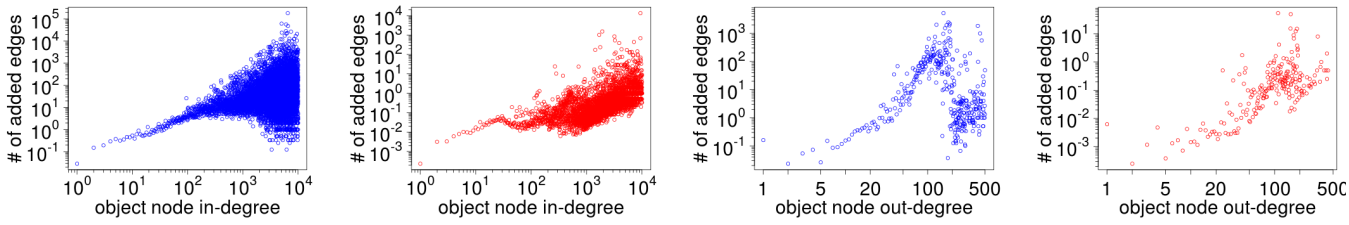
Based on the analyses in Section V, we develop classifiers that verify whether a change is correct or incorrect. We first describe the dataset for the experiment. Subsequently, we summarize the features employed by the classifiers. Thereafter, we describe the classification algorithms. Finally, we introduce the metrics.

A. Dataset

We leverage the Wikidata dataset used in Section V. First, we split the dataset into changes whose object is a URI and changes whose object is a literal. We refer to changes whose object is a URI as the *URI dataset* and those whose object is a literal as the *literal dataset*. The split is due to that different features can be applied to the two datasets. We further divide the two datasets for training and test by time as Heindorf et al. [6] did, since it is unrealistic to use later changes for training to classify earlier changes. Table I provides the description of the dataset. In both datasets, approximately 80% of changes are used for training and 20% for test.

B. Features

Table II summarizes the features used for the classifiers. The first and second columns show the group of features

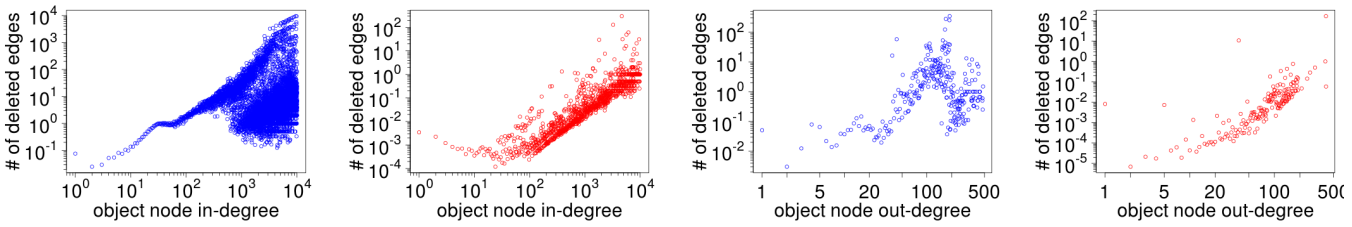


(a) Wikidata: In-degree of correct additions

(b) Wikidata: In-degree of incorrect additions

(c) Wikidata: Out-degree of correct additions

(d) Wikidata: Out-degree of incorrect additions



(e) Wikidata: In-degree of correct deletions

(f) Wikidata: In-degree of incorrect deletions

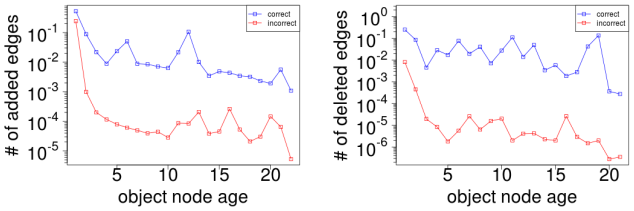
(g) Wikidata: Out-degree of correct deletions

(h) Wikidata: Out-degree of incorrect deletions

Fig. 6: The average degree of object nodes of added and deleted edges. The x-axis shows the average degree of nodes and the y-axis indicates the number of added or deleted edges. Both axes are in logarithmic scale.

TABLE I: The evaluation dataset for training and test.

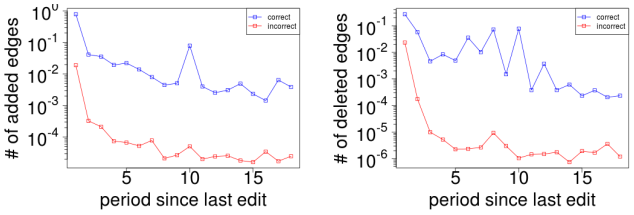
dataset		from	to	# of changes	rate of incorrect changes
URI	training	04/20/2014	01/04/2016	90,234,704	7.63%
	test	01/04/2016	06/21/2016	22,649,334	5.41%
literal	training	04/20/2014	01/04/2016	47,532,819	8.01%
	test	01/04/2016	06/21/2016	8,790,632	4.67%



(a) Wikidata: Added edges

(b) Wikidata: Deleted edges

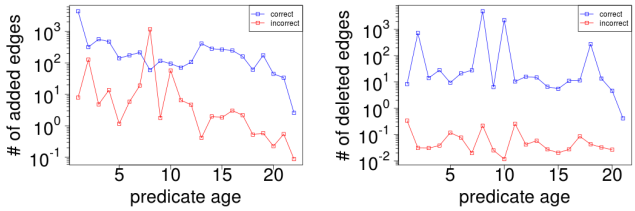
Fig. 7: The average number of added and deleted edges with an object node of age a. The y-axis is in logarithmic scale.



(a) Wikidata: Added edges

(b) Wikidata: Deleted edges

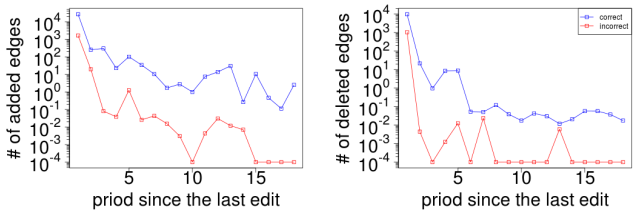
Fig. 8: The average number of added and deleted edges with an object node that is last edited b points in time before. The y-axis is in logarithmic scale.



(a) Wikidata: Added edges

(b) Wikidata: Deleted edges

Fig. 9: The average number of added and deleted edges with a predicate whose age is a. The y-axis is in logarithmic scale.



(a) Wikidata: Added edges

(b) Wikidata: Deleted edges

Fig. 10: The average number of added and deleted edges with a predicate which is lastly used b points in time before. The y-axis is in logarithmic scale.

and the features, respectively. The third and forth columns describe how each feature influences the overall classification performance, which are explained in Section VII. In addition to the features from the analysis in Section V, we employ

"predicate" as a feature, as used by Tan et al. [5] and Heindorf et al. [6]. We convert the predicates to features by one-hot encoding. While the URI dataset uses all 16 features (see Table II), the literal dataset exploits 10 features, since the features from the group "object" are not applicable to the literal dataset. We choose these features, since we wanted to verify whether the simple topological features contribute to assessing KG changes.

TABLE II: Features used by the classifiers for automatic change verification. The third and fourth columns show ROC for the feature ablation analysis described in Section VII.

Group	Feature	URI	literal
subject	in-degree	0.7883	0.6193
	out-degree	0.7915	0.6589
	URI out-degree	0.7884	0.6642
	literal out-degree	0.7946	0.6580
	age	0.7769	0.6310
	last edit	0.9074	0.6393
predicate	age	0.6138	0.4500
	last edit	0.7409	0.6163
	predicate	0.7601	0.6233
object	in-degree	0.7879	-
	out-degree	0.8929	n.a.
	URI out-degree	0.8955	n.a.
	literal out-degree	0.8880	n.a.
	age	0.8240	n.a.
	last edit	0.7713	n.a.
others	flag m	0.7853	0.6510

C. Classification Algorithms and Metrics

Tan et al. [5] observed logistic regression outperforms GradBoost [11] and perceptron [12]. The pilot experiments by Heindorf et al. [6] found out that random forest [13] outperforms logistic regression as well as Naive Bayes. We test all the above-mentioned classifiers and other classifiers including decision tree, support vector machine, and k-nearest neighbor. Our pilot experiment shows that in line with the existing works [5], [6], logistic regression and random forest perform better than the others. Since the results of them are similar, we employ both in the experiment[2]. To avoid overfitting, we use L2 regularization with $\lambda = 0.01$ for logistic regression. We optimize λ by 10-fold cross validation on the training data. For the random first, we optimize the maximal tree depth as 8 by 10-fold cross validation on the training data.

To assess how well the classifiers detect incorrect changes, we use two metrics, the area under curve of the receiver operating characteristic (ROC) and the area under the precision-recall curve (PR), along with Heindorf et al. [6]. While ROC is used to evaluate classification performance in general, PR provides a different view for imbalanced datasets [23]. Please note that we treat incorrect changes as positive class and correct ones as negative class [6]. Thus, precision and recall are defined as the fraction of predicted incorrect changes that are truly incorrect, and the fraction of all truly incorrect changes identified, respectively.

[2]We use the implementations provided by Turi: https://turi.com/products/create/docs/graphlab.toolkits.classifier.html

TABLE III: Result of the change verification on the test data with ROC and PR.

	URI dataset		literal dataset	
	ROC	PR	ROC	PR
logistic regression	0.8350	0.3248	0.6543	0.0116
random forest	0.9183	0.4728	0.4688	0.0043

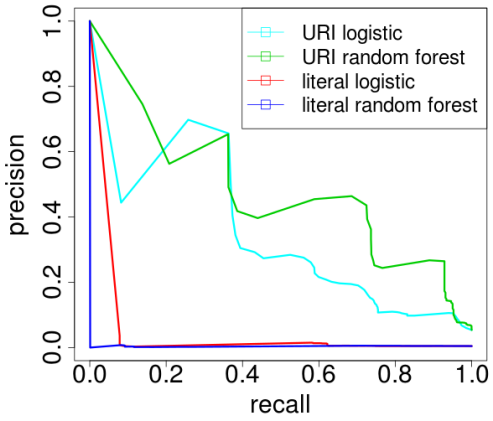


Fig. 11: Precision-recall curves of the classifiers.

VII. Results

First, we report the performance of the change verification methods. Subsequently, we report the influence of each feature.

Table III provides the result of the classification applied on the two datasets. In addition, Figure 11 shows the corresponding precision-recall curves. We observe that the classifiers perform well for the URI dataset, but not for the literal dataset. Since assessing changes whose object is a URI was the most challenging in the previous work [6], we think the novel features can complement the previous works. In terms of the classifiers, random forest outperforms the other on the URI dataset, while logistic regression performs better on the literal dataset. A possible reason of the poor performance for the literal dataset is that literals are inherently not counted as nodes in graphs. Thus, they do not have the evolutionary patterns of the topological features. As observed by Heindorf et al. [6], linguistic features perform much better for the literal dataset.

We individually assess the classification performance using different feature groups. The feature groups "object" and "predicate" perform best for the URI dataset and the literal dataset, respectively. For both datasets, the feature groups "subject" shows the worst performance. It might be a reason why the classifiers do not work for the literal dataset. In order to further assess the influence of each feature, we perform feature ablation analysis, by removing from the classifier one feature at a time, as Tan et al. [5] did. We use random forest for the URI dataset and logistic regression for the literal dataset. The third and fourth columns of Table II show ROC when each feature is not employed. Thus, a smaller value indicates that the feature has a large positive influence. In both

datasets, predicate age has the largest influence. The features relevant to out-degree have the smallest influence, since they are correlated with each other (e .g., out-degree and URI out-degree).

VIII. Discussion

The results of our experiment show that our novel features that take the evolution of the KG into account do improve the state of the art. Our method judges some correct changes as incorrect ones, but does well detect incorrect changes, too. In addition, since we do not use editors' information, our method can be applied to new editors with no history. This is achieved by exploiting for the first time information of the evolution of the KG itself.

Heindorf et al. [6] used the WDVC dataset [21] based on Wikidata for their evaluation. We cannot use this dataset, since the overlap period between the WDVC dataset and the used snapshots is short. Thus, a direct comparison with Heindorf et al. [6] is not possible. Instead, we have used a heuristic from the same authors [21] that labels changes as correct or incorrect, if a change is reverted within four weeks. This heuristic is also used by Tan et al. [5]. In addition, we have conducted a qualitative analysis and manually inspected a random sample of 400 changes in the test data of the URI dataset. Our novel classification heuristics has labeled 23 changes as being incorrect. From those changes, we found that only 1 is a false positive (falsely labeled as incorrect) and 18 false negative. Since the number of false positive is very small, we think that the performance of detecting incorrect changes is properly evaluated. Furthermore, since we used the cross-domain dataset Wikidata, it seems to be reasonable to assume that our idea of exploiting information about the evolution of KGs for change verification can also work on other cross-domain KGs or even on other, domain-specific KGs.

We treat every literal as an indepent node despite two other options: (a) treating lexically identical literals as one node and (b) treating literals that are lexically identical and used by a same predicate as one node. The latter case is motivated by the idea that literals have different semantics depending on the contexts in which they are used. Thus, our results could be biased towards this decision. However, most literals have actually only one incoming edge in both cases. We have investigated this by computing the number of nodes in our setting with the number of nodes in the cases (a) and (b) as follows: $\frac{\text{\# of nodes in the case (a)}}{\text{\# of nodes in our setting}} = 0.85$ and $\frac{\text{\# of nodes in the case (b)}}{\text{\# of nodes in our setting}} = 0.96$. Since the difference between the two cases is low, the influence by how we treat literals is small.

Please note, although automatic verification suggests that there is no human in the workflow, the developed methods are not designed to be run fully without human intervention. In the future, we will investigate how to integrate the developed method into KG editing tool such as Primary Source Tool [4] to facilitate KG editors.

IX. Conclusions

We have analyzed the evolution of KGs, focusing on topological features, using Wikidata. The analysis revealed that the Wikidata KG follows the densification power-law [22]. Subsequently, we analyzed by which kinds of nodes, edges are added or deleted. In summary, the result of the analysis revealed that the KG has similar evolutionary patterns to other graphs. Furthermore, our experiments on change verification demonstrated that the novel topological features are useful to verify incoming changes whose object is a URI.

Acknowledgment

The research described in this paper was supported by the EU H2020 project MOVING under contract no 693092.

References

[1] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga N., D. Gerber, and P. Cimiano, "Template-based question answering over RDF data," in *WWW*. ACM, 2012.
[2] M. Schuhmacher and S. P. Ponzetto, "Knowledge-based graph document modeling," in *WSDM*. ACM, 2014.
[3] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, 2014.
[4] T. P. Tanon, D. Vrandečić, S. Schaffert, T. Steiner, and L. Pintscher, "From freebase to Wikidata: The great migration," in *WWW*. IW3C2, 2016.
[5] C. H. Tan, E. Agichtein, P. Ipeirotis, and E. Gabrilovich, "Trust, but verify: Predicting contribution quality for knowledge base construction and curation," in *WSDM*. ACM, 2014.
[6] S. Heindorf, M. Potthast, B. Stein, and G. Engels, "Vandalism detection in Wikidata," in *CIKM*. ACM, 2016.
[7] H. Paulheim, "Knowledge graph refinement: A survey of approaches and evaluation methods," *Semantic Web*, 2016.
[8] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge Vault: A web-scale approach to probabilistic knowledge fusion," in *KDD*. ACM, 2014.
[9] J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins, "Microscopic evolution of social networks," in *KDD*. ACM, 2008.
[10] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, 1999.
[11] J. Duchi and Y. Singer, "Boosting with structural sparsity," in *ICML*. ACM, 2009.
[12] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Machine learning*, vol. 37, no. 3, 1999.
[13] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, 2001.
[14] M. Nickel, V. Tresp, and H.-P. Kriegel, "Factorizing YAGO: scalable machine learning for linked data," in *WWW*. ACM, 2012.
[15] M. Luther, T. Liebig, S. Böhm, and O. Noppens, "Who the heck is the father of Bob?" in *ESWC*. Springer, 2009.
[16] C. Guéret, P. Groth, C. Stadler, and J. Lehmann, "Assessing linked data mappings using network measures," in *ESWC*. Springer, 2012.
[17] T. Käfer, A. Abdelrahman, J. Umbrich, P. O'Byrne, and A. Hogan, "Observing linked data dynamics," in *ESWC*. Springer, 2013.
[18] R. Dividino, T. Gottron, and A. Scherp, "Strategies for efficiently keeping local LOD data caches up-to-date," in *ISWC*. Springer, 2015.
[19] C. Nishioka and A. Scherp, "Temporal patterns and periodicity of entity dynamics in the linked open data cloud," in *K-CAP*. ACM, 2015.
[20] F. Erxleben, M. Günther, M. Krötzsch, J. Mendez, and D. Vrandečić, "Introducing Wikidata to the linked data web," in *ISWC*. Springer, 2014.
[21] S. Heindorf, M. Potthast, B. Stein, and G. Engels, "Towards vandalism detection in knowledge bases: Corpus construction and analysis," in *SIGIR*. ACM, 2015.
[22] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *KDD*. ACM, 2005.
[23] J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," in *ICML*. ACM, 2006.